Problem 1

(a)

If $n = 10^6$ and $u = 2^{64}$, then Radix sort will make $\lceil \log_b 2^{64} \rceil$ iterations of counting sort. Each Counting sort makes approximately $O(n + b)$ operations; we can neglect the constant factor because it doesn't affect which b is optimal. So we need to minimize

$$(10^6 + b)\lceil \log_b 2^{64} \rceil = (10^6 + b)\lceil 64 \log_b 2 \rceil$$

Using a computer we see that this is minimized at $b = 2^{16} = 65536$ which gives us 4 iterations with approximately 4262144 operations.

(b)

	0	1	2	3	4	5	6	7
h_0	0	0	0	1	1	0	1	1
h_1	0	0	1	0	0	1	1	1
h_2	0	1	0	1	0	1	0	1
h_3	0	1	1	0	1	0	0	1

(c)

This family is not universal. $\Pr\{H(b) = H(c)\} = 3/5 > 1/2$ for H uniformly random over the family.

Problem 2

Create an array $S_{0..u-1}$ where S_i counts the number of occurrences of i. Go through the array once and increment S_i whenever we see i. Now compute P, the prefix sums of S. In essence P_i is one past the last index where values of value i should go. Next create an array I where $I_i = P_{i-1}$ with $I_0 = 0$; each I_i is the index where the next value of value i we see should go. We will maintain an invariant where each segment $[P_{i-1}, I_i)$ will be in the correct indices.

Now go through the original array. Say we're at some index $0 \leq t < n$. Maintain a segment counter s of which what is the largest P_s for which we are currently past. If $t < I_{s+1}$, then we're in a sorted segment; increment t. Else, we need to find a place for the tth element. Say it has value i, While $i \neq s + 1$, i.e. it doesn't belong it in the segment, swap the current element with I_i, and increment I_i. Once the element we swapped it with belongs in our curent segment, increment I_{s+1}. Increment t. If $t = P_{s+1}$, increment s since we're now in the next segment. Repeat.

This is $\Theta(u)$ space since our additional arrays S, P are length u. At each step of the algorithm we swap one more element to the correct position, and don't swap elements once they're in the correct position. So the swapping only incurs an additional $O(n)$ cost at most. So the runtime is still $O(n + u)$.

To see this is correct, we know from recitation that the prefixes correctly determine where elements should go. So when we swap an element of value i to I_i, it is in the correct position; all the segments $[P_s, I_{s+1})$ are in the correct positions. Next we show that the prefix $[0, t)$ is correctly sorted. This is true when $t = 0$. When we increment t, we have a few cases. If $t \in [P_s, I_{s+1})$ for some s, then the tth element is correctly sorted by the above, so $[0, t + 1)$ is also correct. Otherwise, we keep swapping out the current element. Since our inductive hypothesis states that our prefix is correctly sorted, we only swap the current element to future positions, so the prefix is not disrupted and hence still correctly sorted. And we only move on from t when the tth element now belongs in the current segment, and so $[0, t + 1)$ is correctly sorted. So in the end the whole array will be correctly sorted.

Problem 3

(a)

We first prove the following claim: if a is a uniformly random bitstring, then $a \oplus b$ is also uniformly random for any b independent to a. To see this, fix an arbitrary index i and write $c = a \oplus b$.

$$\begin{aligned}\Pr\{c_i = 1\} &= \Pr\{a_i = 0 \wedge b_i = 1\} + \Pr\{a_i = 1 \wedge b_i = 0\} \\ &= \Pr\{b_i = 1 \mid a_i = 0\} \Pr\{a_i = 0\} + \Pr\{b_i = 0 \mid a_i = 1\} \Pr\{a_i = 1\} \\ &= \Pr\{b_i = 1\} \Pr\{a_i = 0\} + \Pr\{b_i = 0\}(1 - \Pr\{a_i = 0\}) \quad \text{(independence)} \\ &= \Pr\{a_i = 0\}/2 + (1 - \Pr\{a_i = 0\})/2 \\ &= 1/2\end{aligned}$$

so c_i, and thus c, is uniformly random.

For the main problem, let us case on if there is a single chunk at which x, y, z all differ. Say first there is at the (chunk) index i. Then

$$h(x) = A \oplus R[x_i][i], h(y) = B \oplus R[y_i][i], h(z) = C \oplus R[z_i][i]$$

for some bitstrings A, B, C where $A \perp R[x_i][i]$, $B \perp R[y_i][i], C \perp R[z_i][i]$ and also there $R[x_i][i], R[y_i][i], R[z_i][i]$ are also independent. So by the lemma we have that $h(x), h(y), h(z)$ are uniformly and independently random, as desired.

Say now there is no single chunk at which x, y, z all differ. Then there must be a chunk where two of them match but a third differs. To see this, since x and y are distinct, there must be an index i for which $x_i \neq y_i$. By the case assumption z_i does not differ from both, so either $x_i \neq y_i = z_i$ or $z_i = x_i \neq y_i$. Without loss of generality suppose that x_i is the odd one out. Then y and z must still be distinct, so there has to be a distinct index j for which $y_j \neq z_j$.

Now, we use the chain rule in a specific order:

$$\begin{aligned}\Pr\{h(x) = v_1 \wedge h(y) = v_2 \wedge h(z) = v_3\} &= \Pr\{h(x) = v_1 \mid h(y) = v_2 \wedge h(z) = v(3)\} \\ &\quad \cdot \Pr\{h(y) = v_2 \mid h(z) = v_3\} \cdot \Pr\{h(z) = v_3\}\end{aligned}$$

We know $\Pr\{h(z) = v_3\} = 1/m$. Now $h(y) = B \oplus R\left[y_j\right][j]$ where B is possibly dependent on $h(z)$ but $R\left[y_j\right][j]$ is independent from both B and $h(z)$. By the lemma $h(y)$ is uniformly and independently random, so $\Pr\{h(y) = v_2 \mid h(z) = v_3\} = \Pr\{h(y) = v_2\} = 1/m$. Finally $h(x) = A \oplus R[x_i][i]$ where A is possibly dependent on $h(y), h(z)$ but $R[x_i][i]$ is uniformly random and independent from $A, h(y), h(z)$. So by the lemma $\Pr\{h(x) = v_1 \mid h(y) = v_2, h(z) = v_3\} = \Pr\{h(x) = v_1\} = 1/m$. So $\Pr\{h(x) = v_1 \wedge h(y) = v_2 \wedge h(z) = v_3\} = 1/m^3$ as desired.

(b)

Let $w = 2$ and $b = 1$. If we denote the values in our table by

Key x	x_0	x_1
Row x_0	a	b
Row x_1	c	d

for $a, b, c, d \in \{0, 1\}$. Note $h(0) = a \oplus b$, $h(1) = a \oplus d$, $h(2) = c \oplus b$. Then

$$h(0) \oplus h(1) \oplus h(2) = a \oplus a \oplus b \oplus b \oplus c \oplus d = c \oplus d = h(3)$$

and so $h(0), h(1), h(2)$ completely determine $h(3)$. So there is no 4-wise independence.